Security Validation Attack Failure on Patched Server Documentation

By Liben Tadele

This section documents how the previously successful NoSQL injection attacks **no longer work** after applying security fixes.

The goal is to **prove** that:

- Vulnerabilities were fixed
- Attacks now fail
- Unauthorized access is blocked

Purpose of This Documentation

This documentation shows that:

- NoSQL injection payloads are rejected
- Authentication bypass is no longer possible
- Sensitive data cannot be extracted
- Role-based access control (RBAC) is enforced correctly

This confirms the application is no longer vulnerable to the demonstrated attacks.

Test Environment

- Server: Fixed version of the NoSQL Injection Vulnerable Server
- Database: MongoDB
- Testing Tool: Postman
- Authentication: JWT
- User Role Tested: Unauthenticated user

Injection Vector 1: Authentication Bypass Attempt

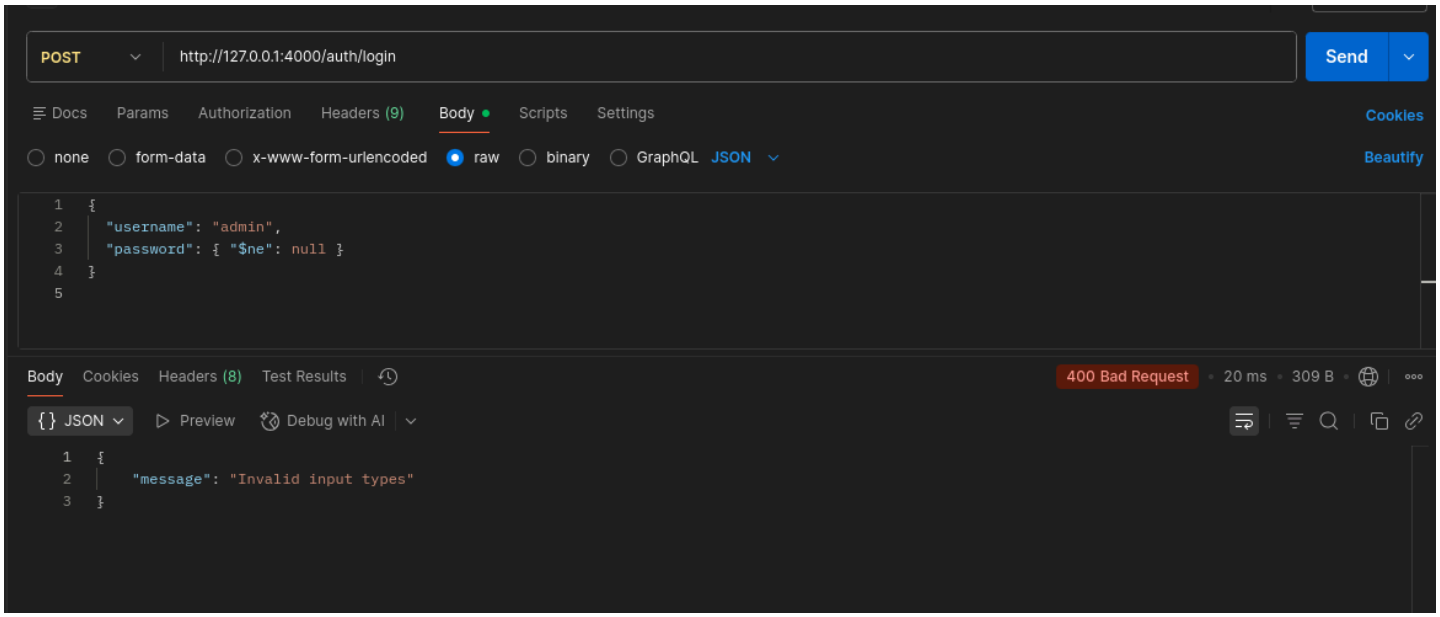
Endpoint Tested

```
POST /auth/login
```

Payload Used (Previously Successful)

```
{
  "username": "admin",
```

```
  "password": { "$ne": null }
}
```



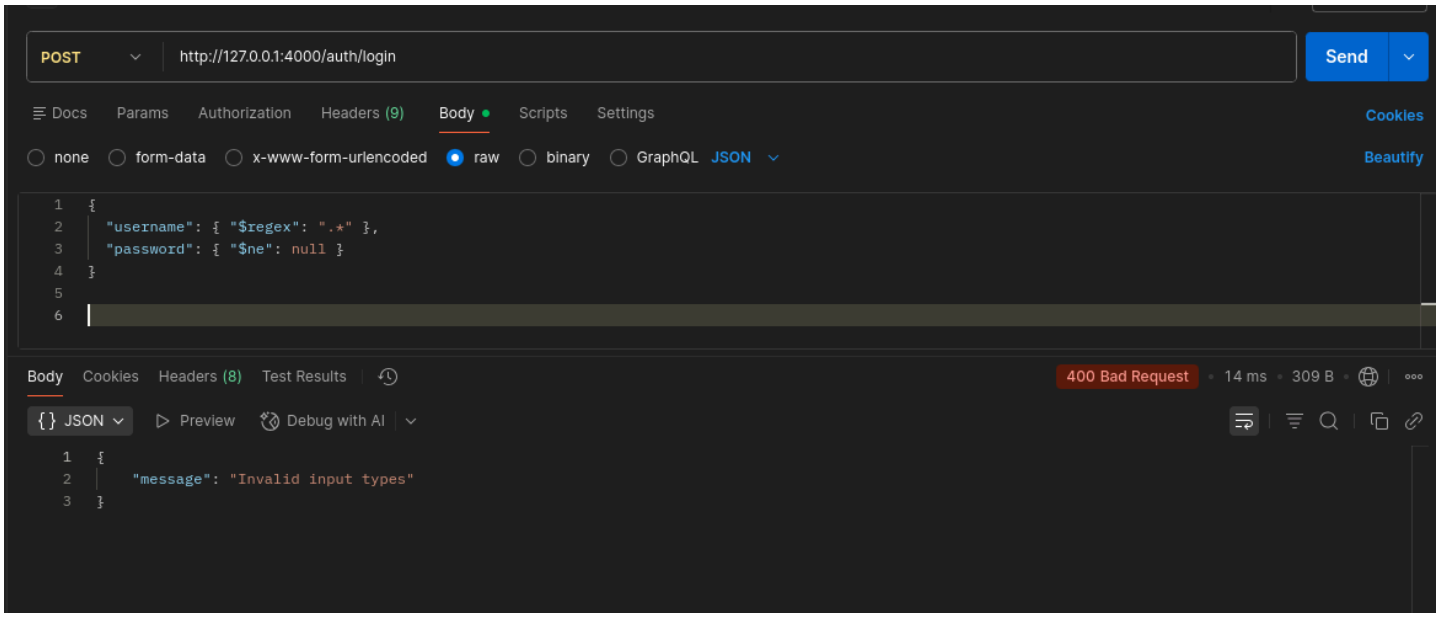
Injection Vector 2: Regex Injection in Login

Endpoint Tested

`POST /auth/login`

Payload Used

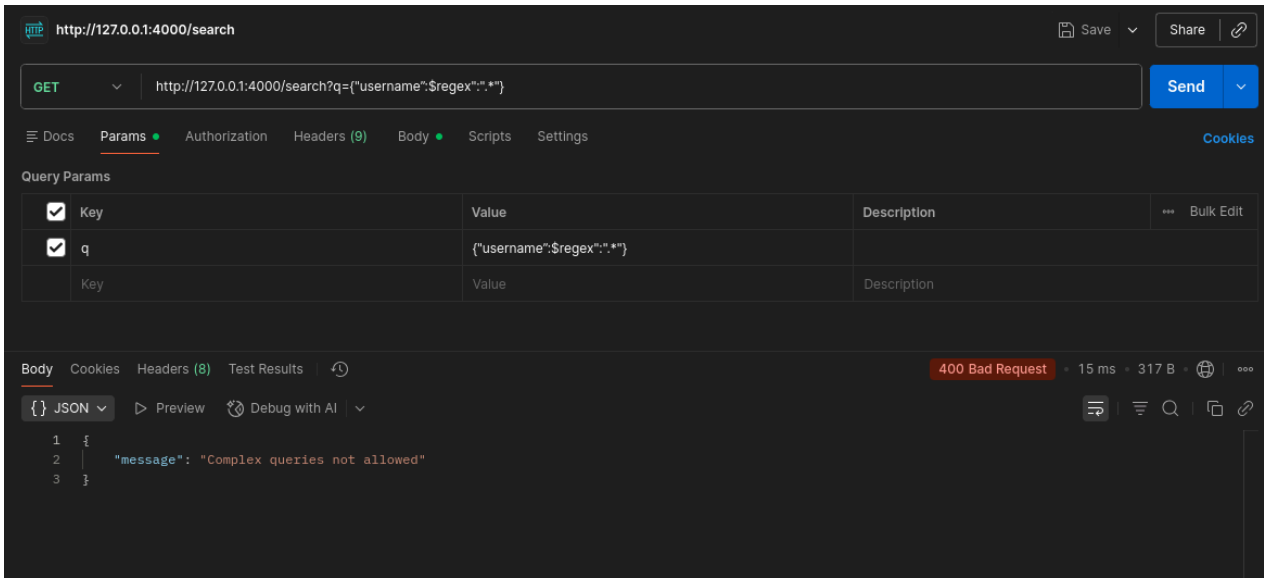
```
{
  "username": { "$regex": ".*" },
  "password": { "$ne": null }
}
```



Injection Vector 3: User Enumeration via Search

Endpoint Tested

`GET /search?q={"username":$regex":".*"}`



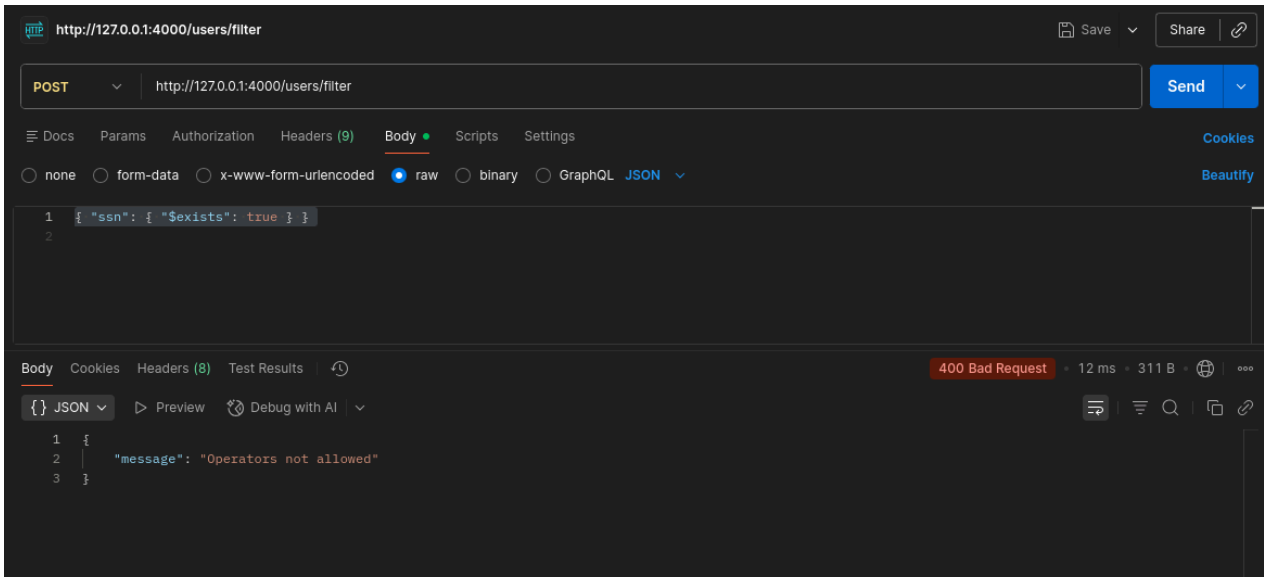
Injection Vector 4: Sensitive Data Extraction

Endpoint Tested

`POST /users/filter`

Payload Used

```
{ "ssn": { "$exists": true } }
```



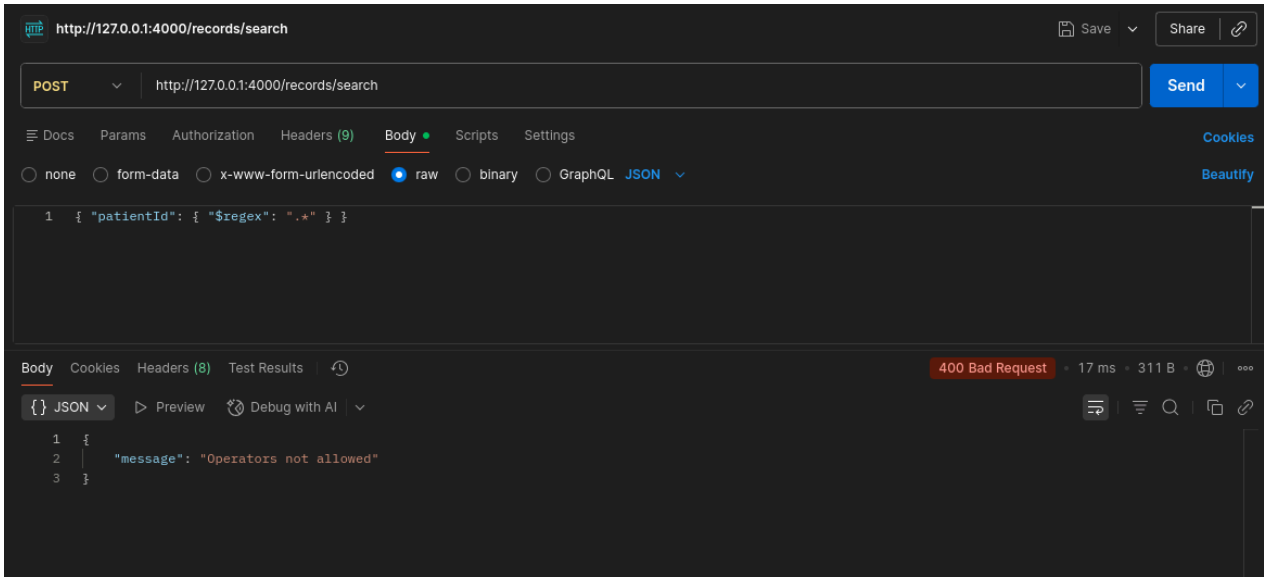
Injection Vector 5: Medical Records Extraction

Endpoint Tested

```
POST /records/search
```

Payload Used

```
{ "patientId": { "$regex": ".*" } }
```



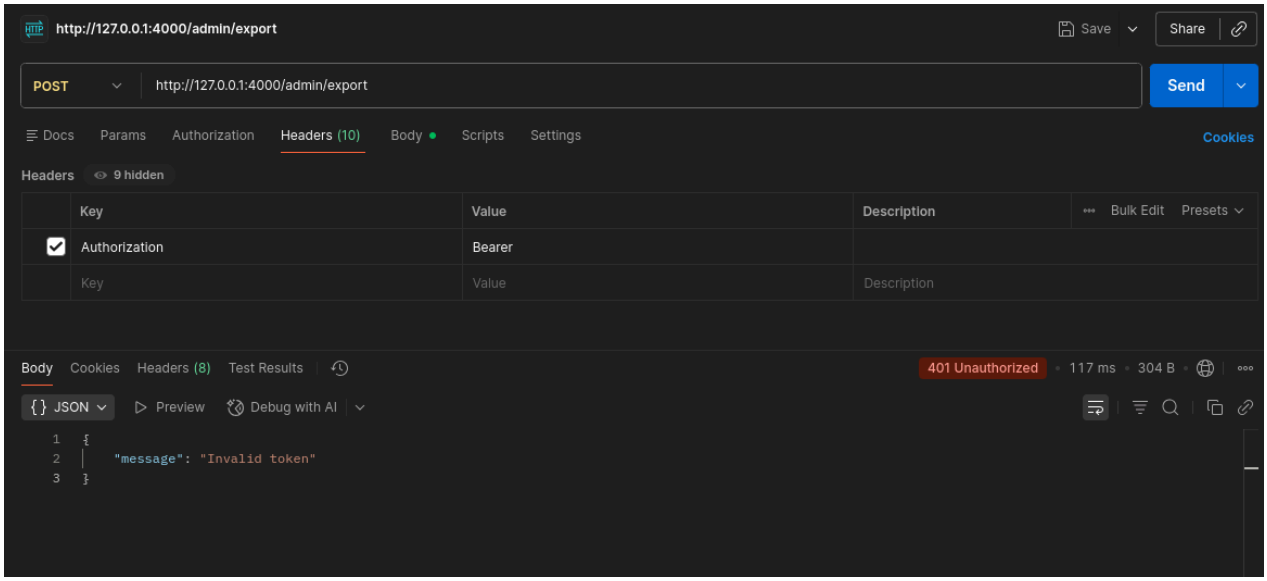
Injection Vector 6: Admin Endpoint Access Without Proper Authorization

Endpoint Tested

`POST /admin/export`

Test Condition

- No JWT token
- Invalid or non-admin token



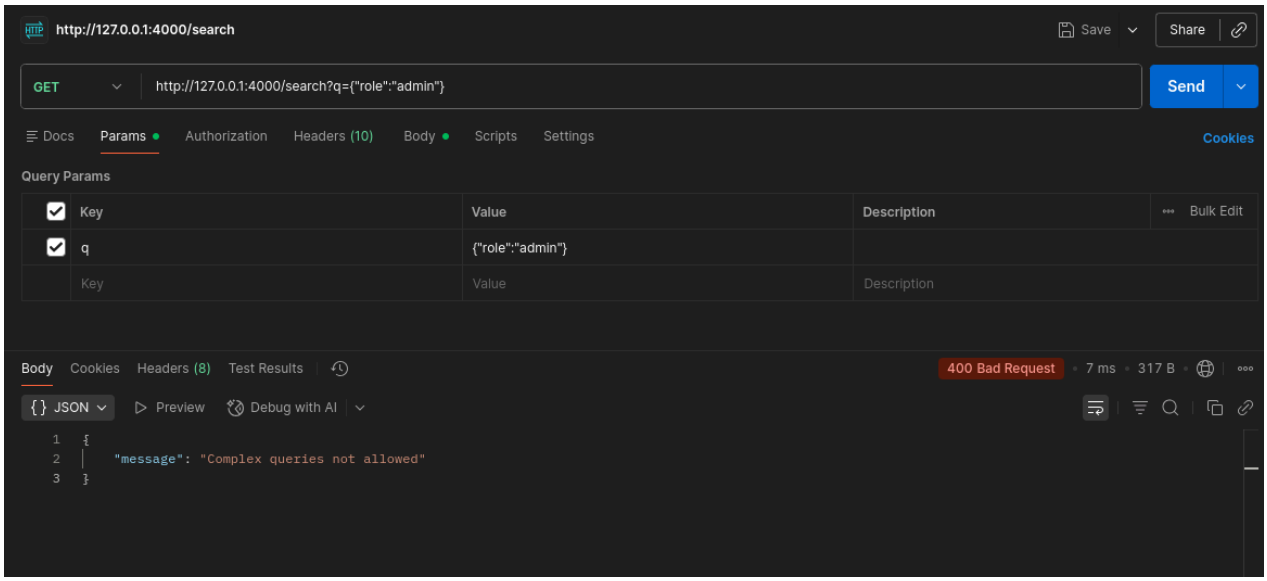
Injection Vector 7: Extract Admin Accounts

Endpoint

`GET /search`

Payload

`?q={"role":"admin"}`



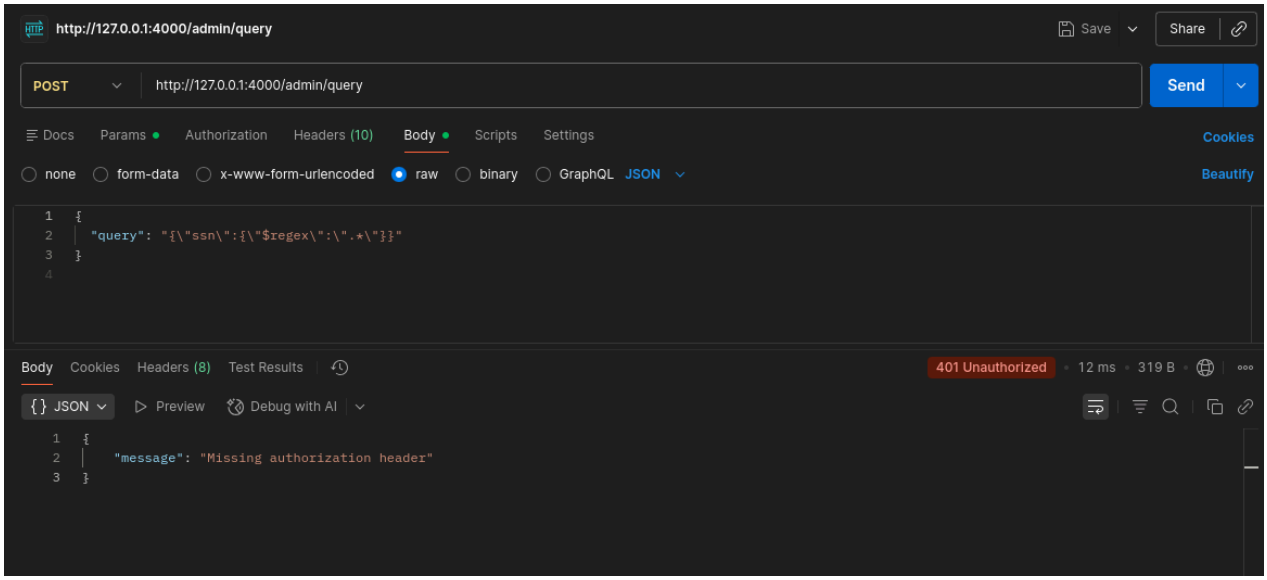
Injection Vector 8: Admin Query Injection

Endpoint

`POST /admin/query`

Payload

```
{ "query": "{\"ssn\":{\"$regex\":\".*\"}}" }
```



Automated Exploit Script Validation

An automated Python exploit script that previously succeeded in exploiting the NoSQL injection vulnerabilities was re-executed against the fixed server.

Exploit Script Used

- Script: `nosql_exploit.py`

- Purpose: Automate authentication bypass and full data extraction

Security Validation Outcome

All tested NoSQL injection vectors that previously resulted in authentication bypass, data exposure, or privilege escalation were successfully mitigated. The application now correctly treats all user input as untrusted data and prevents it from being interpreted as database logic.

Validation testing confirms that malicious MongoDB operators are blocked, unauthorized requests are denied, and sensitive resources are accessible only to properly authenticated and authorized users. No evidence of data leakage or privilege abuse was observed during post-fix testing.

This outcome demonstrates that the applied remediation measures are effective and that the system meets basic security expectations for input validation, authentication, and access control.